MIDTERM EXAM I SOLUTIONS
CSCI 61: DATA STRUCTURES
SPRING 2016

1. (10 points) Simulate step-by-step the function `partition()` on the following array:

   `9 1 8 3 7 6 5 2 10 4`

   Show the array after each swap.

   Answer:

   ```
   9 1 8 3 7 6 5 2 10 4
   1 9 8 3 7 6 5 2 10 4
   1 3 8 9 7 6 5 2 10 4
   1 3 2 9 7 6 5 8 10 4
   1 3 2 4 7 6 5 8 10 9
   ```

2. (10 points) What is the output of the following program:

```
#include <iostream>
#include <queue>

using namespace std;

int main()
{
        queue<int> q;
        for (int i = 1; i <= 7; ++i)
                q.push(i);

        int n(0);
        while (q.size() != 1)
        {
                int front = q.front();
                q.pop();
                if (n++ % 3 != 0)
                        q.push(front);
        }
        cout << q.front() << endl;

        return 0;
}
```

Answer:

```
1 2 3 4 5 6 7
  2 3 4 5 6 7
    3 4 5 6 7 2
      4 5 6 7 2 3
        5 6 7 2 3
          6 7 2 3 5
            7 2 3 5 6
              2 3 5 6
                3 5 6 2
                  5 6 2 3
                    6 2 3
                      2 3 6
                        3 6 2
                          6 2
                            2 6
                              6 2
                                2
```

```
Output = 2
```

3. (10 points) Consider the following sorting algorithm:

```
// pre: none
// post: a[0..n-1] are sorted in nondecreasing order
template <class T>
void mystery_sort(T a[], int n)
{
    for (int i = n-1; i > 0; --i)
        for (int j = 0; j < i; ++j)
           if (a[j+1] < a[j])
              swap(a[j+1], a[j]);
}
```

(a) Simulate this algorithm step-by-step on the array `3 1 4 5 2`. Show the array after each swap.

Answer:

```
3 1 4 5 2

1 3 4 5 2
1 3 4 2 5

1 3 2 4 5

1 2 3 4 5
```

(b) On arrays of size n, how many times is the `if` statement executed ?

Answer:

$$(n-1) + (n-2) + \cdots + 3 + 2 + 1 = \frac{n(n-1)}{2}.$$

(c) Is this sorting algorithm stable ? Justify your answer.

Answer:

Yes, because a swap is performed only when an element is strictly greater than its right neighbor.

4. (10 points) Evaluate the following postfix/prefix expressions:
   (a) `100 4 2 / / 18 9 / / 7 9 3 / * 5 6 1 - + * +`
   Answer: 235
   (b) `+ / + 1 5 - + 5 7 9 * * 3 3 / + 8 6 * 1 2`
   Answer: 65

5. (10 points) Write a function

   `string postfix_to_full(const string & postfix)`

   to convert a postfix expression to the equivalent **fully parenthesized** infix expression.

   Assume that operands and operators are separated by at least one space; also assume that no spaces separate an operand from its positive/negative sign, if present.

   For example, given expression `3 4 + 5 *`, the output should be `( ( 3 + 4 ) * 5 )`.

   Answer:

```
string postfix_to_full(const string & postfix)
{
        istringstream ss(postfix);
        string token, left, right;
        stack<string> s;

        while (ss >> token)
        {
                if (token == "+" || token == "-" || token == "*" ||
    token == "/" || token == "%")
                {
                        right = s.top();
                        s.pop();
                        left = s.top();
                        s.pop();
                        s.push("( " + left + " " + token + " " + right + " )");
                }
                else
                        s.push(token);
        }
        return s.top();
}
```